## I. RÉSOUDRE LE PROBLÈME DU STOCKAGE

Pour le stockage des données et métadonnées de nos NFT Book nous avons fait le choix de l'utilisation du stockage décentralisé par rapport au stockage centralisé pour les raisons suivantes:

**- Résilience :** Les données sont réparties sur plusieurs nœuds, ce qui réduit le risque de perte en cas de défaillance d'un serveur unique.

**- Sécurité renforcée :** Les données sont souvent chiffrées et dispersées, rendant leur accès non autorisé très difficile.

**- Contrôle total :** Les utilisateurs conservent la maîtrise de leurs données, garantissant leur intégrité et évitant toute suppression ou modification par une entité centralisée.

**- Immutabilité :** Une fois stockées, les données ne peuvent pas être altérées, assurant la traçabilité et l'authenticité des NFT.

**- Indépendance Vis-à-Vis des Fournisseurs :** L'utilisation de solutions décentralisées diminue la dépendance à des services centralisés, réduisant les risques associés à leur gestion.

Parmi la multitude de stockages décentralisés nous avons fait le choix du **protocole IPFS** (InterPlanetary File System).

IPFS est un protocole d'hébergement décentralisé de données. Son rôle est de connecter un ensemble de serveurs indépendants les uns des autres, pour qu'ils se partagent des fichiers sans dépendre d'un serveur central. Le protocole a été créé en 2015 par Juan Benet, et c'est une technologie de référence du web décentralisé (dWeb).

Pour en savoir plus sur IPFS rendez sur ce site:
https://cryptoms.fr/technologie/2021/01/27/ipfs-l-hebergement-de-donnees-en-pair-a-pair.html

## II.DÉMO SUR IPFS

Après avoir installé les logiciels, configurations et connection de votre nœud avec celle du réseau mondial pour assurer que vos données sont en permanence stockées quelque part.

Suivre la documentation pour les configurations:

https://docs.ipfs.tech/install/command-line/#determining-which-node-to-use-with-the-command-line

https://docs.ipfs.tech/how-to/command-line-quick-start/#prerequisites

Nous allons stocker un livre pdf sur le réseau à travers notre noeud:

Lancer le démon IPFS : Avant d'ajouter des fichiers, nous allons nous assurer que notre démon IPFS est en cours d'exécution. Pour démarrer le démon IPFS, exécutons la commande suivante dans PowerShell :

**ipfs daemon**

Puis ensuite stocké votre document, sur powershell tapez la commande:

**ipfs add "chemin de notre document"**

```
PS C:\Windows\system32> ipfs add "C:\Python&Django.pdf"
 53.88 MiB / 53.88 MiB [=============================================
added QmU1KHKGkVB3hcXJK8Xgox4dr9PC5E8XG5124Ri5fg17Us Python&Django.pdf
 53.88 MiB / 53.88 MiB [=============================================
PS C:\Windows\system32>
```

Lorsque vous téléchargez votre fichier PDF sur IPFS, il vous renvoie un hash unique CID (Content Identifier). Ce hash représente l'emplacement du fichier sur le réseau IPFS et garantit que le fichier est accessible publiquement (pour tout ce qui possède ce hash) de manière décentralisée.

Pour nous ce hash correspond à:
**QmU1KHKGkVB3hcXJK8Xgox4dr9PC5E8XG5124Ri5fg17Us**

Nous pouvons afficher le contenu de notre document par la commande:

**ipfs cat /ipfs/QmU1KHKGkVB3hcXJK8Xgox4dr9PC5E8XG5124Ri5fg17Us**

Nous pouvons également accéder à ce fichier via un gateway IPFS dans un navigateur web en utilisant une URL comme celle-ci :

https://ipfs.io/ipfs/<CID> ou <CID> représente le hash de notre document, dans notre cas on aura:

https://ipfs.io/ipfs/QmU1KHKGkVB3hcXJK8Xgox4dr9PC5E8XG5124Ri5fg17Us

Les personnes à qui vous partagez ce lien peuvent simplement le coller dans leur navigateur, et le portail IPFS (comme ipfs.io) récupérera et affichera le fichier à partir du réseau IPFS.

**NB:** si vous rencontrez des erreurs d'affichage du document cela voudrais dire simplement que soit votre noeud est hors ligne(vous pouvez le vérifier par la commande **ipfs daemon**) ou bien  le noeud a l'origine du stockage n'est pas connecté aux autres noeuds du réseau pour une accessibilité universel.

Vous pouvez remédier à cela à travers les solutions suivantes:

- Si vous êtes bien connecté mais avez peu de connexions à d'autres nœuds, vous pouvez essayer d'étendre le réseau en vous connectant à plus de pairs, par la commande suivante:

**ipfs swarm connect <peer-multi address>**

ou **<peer-multi address>** représente le hash d'autres noeud du réseau

- Si vous voulez garantir que votre fichier est toujours accessible, vous pouvez utiliser un service de "**pinning**" public (comme Pinata) pour maintenir une copie permanente de votre fichier sur des serveurs publics IPFS ( c'est payant ).

Dans notre cas nous avons utiliser une autre approche, pour des raisons de coût et de rapidité nous allons travailler en local c'est à dire nous avons fait de sorte que seul les personnes qui sont connectés à votre réseau wifi et qui possède le hash du document pourront le lire, pour cela nous allons ouvrir le fichier de configuration d'IPFS, dans la section adress aulieu de 127.0.0.1 nous mettons

0.0.0.0 et nous téléchargeons l'extension **IPFS companion**, cela nous permet d'utiliser IPFS dans notre réseau wifi sans tenir compte des contraintes cités ci-dessus.

Dans ce cas pour avoir accès à notre document Python&Django.pdf

http://192.168.1.3:8080/ipfs/QmU1KHKGkVB3hcXJK8Xgox4dr9PC5E8XG5124Ri5fg17Us

192.168.1.3 correspond à l'adresse ipv4 de l'ordinateur qui héberge le nœud sur lequel le document est sur IPFS

**Récapitulatif :**

Nous avons fait le choix de l'utilisation du stockage décentralisé pour stocker nos documents, parmi les choix des stockages décentralisées qui existent nous avons préféré le protocole IPFS.

Et nous avons installé des logiciels, le configurer et connecter à votre nœud au reste du réseau.

Nous avons appris également à stocker nos documents et à faire en sorte qu'ils soient disponibles permanent en utilisant des techniques comme la connexion au maximum de notre nœud au reste du nœud du réseau mais également en utilisant le "pinning".

**PROBLÈME:**

Vous avez sans doute remarquez que lors de notre présentation précédente, toute personne ayant le hash ou le CID de notre document pouvait avoir accès au contenu et même le télécharger.

Dans les méta-données inscrite dans les NFT se trouve alors le CID pointant vers notre pdf or le problème est que, comme la blockchain est public toute personne pourrait lire et récupérer ce hash ainsi pouvoir avoir accès au contenu du document même s'il n'en est pas le propriétaire du NFT.

Ce qui pose un problème fondamental à la solution qu' on veut résoudre.

## III. SOLUTIONS ET CONCEPT

Pour éviter que n'importe qui puisse accéder aux contenus d'un document s'il n'a pas le NFT associé à ce document nous allons procéder comme suit:

- **Chiffrer le document** avant de le mettre sur le réseau du protocole IPFS (de telle sorte qu'une personne même s'il a le document ne puisse voir son contenu sans avoir la clé de chiffrement.

- **Stocker les clés de déchiffrement** de tout les documents dans un serveur(ou API) très sécurisé.

- **Créé maintenant des NFT** associé à chaque document en utilisant le hash ( tokenURI) de la métadonnée du document

- **Créer un smart contrat pour vérifier** si une personne possède un NFT associé à un document, si telle est le cas l'API autoriserait l'utilisation de la clé de déchiffrement ( sans la divulguer) pour que l'utilisateur puisse lire le contenu du document.

Ainsi comme la clé de déchiffrement n'est pas stockée sur la blockchain, juste le tokenURI des métadonnées cela évitera que n'importe qui puisse lire le contenu du document, seul les détenteurs du NFT pourraient lire le contenu. On aurait donc résolu ce problème.

## IV. MISE EN PLACE DES SOLUTIONS PAR LA CRÉATION D'UN NTF ASSOCIÉ À UN DOCUMENT

Pour se faire nous allons suivre les étapes suivantes:

### 1. Chiffrer le document

Nous allons suivre chaque pdf avant de le stocker sur IPFS,

Nous allons utiliser une commande avec **OpenSSL** qui sert à chiffrer de de fichier avec l'algorithme **AES-256** en mode **CBC** (Cipher Block Chaining).

commande :

**openssl enc -aes-256-cbc -salt -in "Python&Django.pdf" -out "C:\Users\balma\Documents\fichier_chiffre.pdf" -pbkdf2 -pass pass:mot_de_passe**

"Python&Django.pdf": peut représenter le chemin de notre répertoire du fichier à chiffrer mais si nous sommes déjà dans ce répertoire il suffit simplement de mettre le nom du fichier.

mot_de_passe : c'est la clé sur laquelle il nous faudra pour déchiffrer (ou lire) le fichier chiffrer

Pour le test cette clé sera donnée de façon volontaire, mais dans le cadre du déploiement il nous faudrait générer des clés en utilisant un algorithme où même l'administrateur de la plateforme ne pourrait le voir seul le smart contract.

**2. Création d'un smart contrat qui stockera la clé de déchiffrement:**

Ainsi ce contrat intelligent pourra vérifier si vous possédez réellement un NFT associé à un quelconque document, si oui le contrat déchiffre le document en question pour que vous puissiez le lire.

**3. Déposer le document chiffrer sur IPFS**

Après le dépôt **voici le hash du document:**

**4. Déposer une image de notre livre sur IPFS**

Voici l'adresse pour accéder à l'image sur IPFS:
http://192.168.1.3:8080/ipfs/Qmf6dtfa8deG9CH4VPRM9dgyCuHjGo6ruYrxqgFFRYJGUX

**5. Création du méta donnée correspondant au document en question**

Ce méta donnée est un fichier .json contenant les infos suivantes:

{

"name": "Python&Django",

"description": "Principes et bonnes pratiques pour les sites web dynamiques.",

"image":
"http://192.168.1.3:8080/ipfs/Qmf6dtfa8deG9CH4VPRM9dgyCuHjGo6ruYrxqgFFRYJGUX",

"pdf":
"http://192.168.1.3:8080/ipfs/QmU1KHKGkVB3hcXJK8Xgox4dr9PC5E8XG5124Ri5fg17Us"

}

**6. Déposer le fichier meta_data.json sur IPFS**

Après ce dépôt nous obtenons ce hash:
**QmVtBf1R8H9WXwgZpH537CVu6VHUJe9Nt6Cyq2YEKqNQf8**

Ce hash correspond également à ce qu'on appelle tokenURI et servira à minter le NFT associé au document pdf.

Pour avoir accès au contenu du meta donnée meta_data.json
http://192.168.1.3:8080/ipfs/QmVtBf1R8H9WXwgZpH537CVu6VHUJe9Nt6Cyq2YEKqNQf8

**7. Création du NFT pour notre document**

Nous allons faire appelle à notre fonction BookMinted présent dans le code solidity sans oublier à introduire le hash du méta donné connu sous le nom de tokenURI.